H

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/458,121	12/08/1999	GAL MOAS	042390.P7162	8466

7590 08/16/2004

JOHN P WARD
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD 7TH FLOOR
LOS ANGELES, CA 90025

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 08/16/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/458,121	Applicant(s) MOAS ET AL.
	Examiner Tuan A Vu	Art Unit 2124

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 May 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4-11,13-20 and 22-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4-11,13-20 and 22-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08 December 1999* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 5/23/2004.

As indicated in Applicant's response, claims 1 and 19 have been amended. Claims 1, 2, 4-11, 13-20, and 22-25 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claims 1 and 19 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1 and 19 recite the limitation "said block of code" in line 3, 6, respectively. There is insufficient antecedent basis for this limitation in the claim; and 'said block of code' will be treated as 'a block of code'.

Appropriate correction is required.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-2, 4-7, 10-11, 13-16, 19-20, and 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Benson, USPN: 5,301,325 (hereinafter Benson), in view of Gosling, USPN: 5,668,999 (hereinafter Gosling).

As per claim 1, Benson discloses a method of monitoring processor resources, such method comprising:

testing blocks of code (e.g. *nodes, flow graph* - col. 12, line 21 to col. 13, line 47), the blocs of code including multiple instructions (e.g. *flow graph ...link... tuples into node or blocks* - col. 11, lines 11-22; node *4, 5, 6* – Fig. 8; *block begin, block end* – Fig. 4; Fig. 5 --Note: Processing the flow graph by visiting nodes represented each by tuple structures, each tuple storing a block begin and block end, is equivalent to blocs of code including multiple instructions)

to determine at the start of said block of code if the resources of an architectural stack are correctly alloted for the block (e.g. *stack depth ... with the node* - col. 13, lines 47 to col. 14, line 5; Fig. 12) for the instructions to be executed; and

signaling an error if said resources needed for block of code are not correctly allotted (e.g. col. 4, lines 61-66; col. 13, lines 48-57).

But Benson does not explicitly specify that the testing at the start of each block of code is for determining if the architectural stack resources needed by such block of code are available; nor does Benson expressly specify that signaling an error is in response to said stack resources (needed for the block of code) not being available. However, Benson discloses processing of block of code since starting entry point until exit points in the flow graph to determine stack depth discrepancy (e.g. col. 18, lines 13-19; Fig. 12), checking whether input/output registers are set at the beginning of head of a routine (e.g. Fig. 11); hence has taught the requirement to see to it that references needed by the block of code internal instructions for being correctly set or adjusted (col. 13, lines 28-42; Fig. 16) for execution as well as stack depth allocation (i.e. available resources) for correct

execution of instructions or allocation registers needed for a procedure. The checking to verify whether data pushed in the stack are actually available for the runtime execution of code during a pre-runtime verification is further evidenced with Gosling's disclosure. Indeed, Gosling, in a similar method to verify program code to learn about stack information and changes using analogous checking and error notifying as Benson, discloses emulating the runtime stack in order to determine whether the virtual operands, variables, or instructions, i.e. resources needed for runtime code are correctly matched with a previously recorded snapshot of the emulated stack and generate error messages when they aren't matched (e.g. cols.14-16, Appendix 1; Fig 4A-G; step *440* – Fig. 4B); hence has suggested how to check if the resources that will be needed for the runtime are available and correctly so. Based on Benson's intention to ensure that stack allocating of memory would not prevent correct execution of a block of code as mentioned above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the graph node traversal and checking by Benson with the matching of runtime needed stack resources with pre-runtime emulated stack data, and generate error message when those resources are not available as taught by Gosling because this would enhance the method of Benson with timely integrity checking of code prior to runtime; and in so doing preclude stack overflow, and/or accommodate for data and platform discrepancies between environments in which the program is to be executed (see Gosling: col. 1, line 33 to col. 2, line 39).

As per claim 2, Benson discloses determining a set of available resources for each block of code and check the correctness of stack allocation of such resources (re claim 1) but does not explicitly specify determining that a set of such resources will be

available after said block of code has been executed. However, Benson mentions about establishing resources state after the block of code has been executed (e.g. col. 13, lines 28-40) and what next block of code needs to have checked (e.g. Fig. 16). In view of such systematic block of code traversal and in combination with the teachings by Gosling, the limitation as to determine the set of needed resources available after the previous block of code has been executed would have been obvious for the same rationale as set forth in claim 1 above; and because the checking cannot stop at one block of code to ensure the integrity of the runtime data when in opposite, all the blocks of code are to be verified.

As per claim 4, Benson discloses compile time (*similar to a compiler* – col. 4, lines 1-38).

As per claim 5, Benson does not explicitly disclose dynamically determining the resources; but Gosling discloses the verifier to be a dynamic program to check stack proper manipulations (e.g. col. 4, lines 47-59). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the block of code checking by Benson to include the determining of runtime needed stack resources as taught by Gosling using a dynamic verifier because this would enable Benson code to be byte code usable in byte code interpreting environments as mentioned by Gosling, thus extending the code applicability of Benson's product to more executing environment.

As per claims 6 and 7, Benson does not specify branching to a handler routine but Gosling discloses a exception handler routine (e.g. col. 12, lines 16-18); and it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement such handler routine to Benson's method detects a dire fault in resources allocation found in the stack especially when Benson's application program is byte codes

used in a dynamic environment like that suggested by Gosling in order to address data incompatibility issue in a dynamic manner without interrupting the application process flow, thus obviating extraneous recovery network or business resources. But in case of simulation as in Gosling' s teaching, simulating an exception handling routine would also have been obvious in case the whole process of verifying resources is for a pre-runtime environment.

As per claims 10-11, and 13-16, these claims are the computer-readable medium (see Benson: disk 17 -Fig.2) or apparatus claims corresponding to method claims 1-2 and 4-7, respectively, hence are rejected herein with the same reasons as set forth above.

As per claim 19, Benson discloses a computer readable medium having a first set of instructions (e.g. *front-end ...input language, input to the translator* – col. 8, lines 39-56), which when executed, generate a second set of instructions through a binary translation process, the second set of instructions (e.g. *intermediate code* – col. 11, lines 8-22; Fig. 4) when executed cause the processor to perform a method comprising the steps of:

testing at the start of a block code (...resources available); and

signaling (...resources not available) just as have been recited in claim 1 above.

Hence these step limitations are rejected using the corresponding rejection of claim 1 as set forth therein, including the rationale with the obvious motivation to combine Benson and Gosling.

As per claims 20 and 22-24, these claims are the computer-readable medium claims corresponding to method claims 2 and 5-7, respectively, hence are rejected herein with the same reasons as set forth above.

6. Claims 8-9, 17-18, and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Benson, USPN: 5,301,325 and Gosling, USPN: 5,668,999; as applied to claims 1, 10, and 19, respectively; and further in view of Yellin et al., USPN: 5,740,441(hereinafter Yellin).

As per claim 8, Benson does not specify using bit vector to represent resources but does provide condition code and tuple to represent resources in a flow graph tree (e.g. Figs. 5-6) while Gosling discloses using exception handler but in a method to pre-verify correctness of data prior to runtime using snapshot of stack to emulate runtime data requirements similar to that of Gosling, Yellin discloses using bit vector to implement the jump subroutines with bit vector to expedite the reaching to subroutine address (e.g. col. 6, lines 6-23; Fig. 3). To implement a dynamic checking as mentioned in claim 6 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a bit vector as taught by Yellin to the resources investigated from the stack in Benson/Gosling's combination to expedite the process of traversing the flow of block of code for verifying of routines, thus to maintain the dynamic resources integrity checking and averting usage of error recovery resources during a runtime environment (e.g. byte code in JIT machine) where possibly processor and/or volatile resources would be limited, such potential limitation being well known in remote devices (e.g. wireless devices) in which the distributed byte codes as taught by Gosling are downloaded and executed.

As per claim 9, Benson and Gosling do not mention about bit vector; but Gosling mentions about simulating in a dynamic environment and using exception handling (re claims 5 and 6). Further, in view of Yellin's teaching of bit vector in a stack emulation

environment similar to Gosling, the limitation as to generate a bit vector dynamically would also have been obvious for the same reasons as mentioned in claim 8 above.

As per claims 17-18, these claims are the computer-readable medium claims corresponding to method claims 8-9 above, respectively, hence are rejected herein with the same reasons as set forth above.

As per claim 25, this claim is the computer-readable medium claim corresponding to method claim 8, and is rejected herein with the same reasons as set forth above.

Response to Arguments

7. Applicant's arguments filed 5/23/2004 have been fully considered but they are not persuasive. Following are the most representative arguments raised by Applicants and the corresponding responses by Examiner.

(A) Applicants have submitted that a Prima Facie must be established (Appl. Rmrks, pg. 7, middle para), and that Benson only discloses tracking stack usage within resources and return address references from valid local references (Appl. Rmrks, pg. 8, top 3 para). The rejection points to specific parts of Benson in which at the entry point to each node being visited a stack depth is checked for sufficient allocation; and this substantially reads on the claim's limitation as interpreted by Examiner. Indeed, the limitation as to testing at the start of each block of code, to determining if resources of an architectural stack that are needed by a block of code are available is substantially, if not entirely, covered by Benson's checking of stack depth (see Fig. 12) when starting to analyze a node wherein each tuples are further checked for correct references allocation. The fact that more checks at the tuple level are made after the stack depth (see Fig. 16) has been

tested only affirms to the fact the resources needed (via the tuples) inside the block of code can be equated to the stack allocation required for that portion of block at the start of which the stack-depth test has been made. Since Benson does not expressly disclose the availability of resources for the block of instructions to be executed, Gosling has been brought in to enhance Benson's intended method of checking stack correctness while traversing flow graph or blocks of code. The claim as recited and interpreted does not preclude nor does it impose the absence of the step of tracking stack usage as brought up by Applicants. In view of the above observations, the portions cited in the rejection are aimed at addressing the claim limitation as interpreted. The motivation to pre-arrange and test stack resources usage as taught by Gosling's emulation method has been set forth in the rejection as Gosling is to enhance the purpose that is apparently not explicitly worded from Benson's cited sections. In regard to which, Benson's intentions and teachings clearly convey the concept of checking correct resources availability (in a sense of correct stack resources being available) when tracing blocks of code in association with stack and register allocation as has been pointed out in the rejection. The rejection combines the stack allocation checking upon entering a block by Benson with the pre-usage stack emulation functions for availability checking by Gosling; with the motivation such as to provide useful first hand knowledge of resources prior to actual execution. Hence, a case of prima facie has been established.

(B) Applicants have submitted that Benson teaches detection so that 'user can be advised … appropriate changes to source code' and that Benson is processing each instruction individually (Appl. Rmrks, pg. 8, bottom 3 para). While it is true that Benson's system detects stack changes and that the processing of reference addreses is

effected at the tuple level, the overall flow graph traversal by Benson does approach each block of code or tree node from an entry point, and thereupon, the checking of stack resources needed for that node is also one of Benson's step; and this has been addressed in section A above. As long as the cited sections by Benson address the limitation as amended from claim 1, the points raised concerning the other aspects of Benson's method become moot. Further, the checking at the beginning of a node and subsequent checking when processing inside each block as taught by Benson do not read away from what is interpreted by Examiner when reading the claim. At the start of each block, a test has been made to find out if stack resources for that particular block are correctly set for execution of that block. The rationale as to combine Benson's intention with Gosling's emulation method has been set forth in the rejection. Applicants while pointing other aspects of Benson's method, actually fail to point out how the combined teachings by Benson and Gosling teach away from claim 1 limitation; nor are applicants able to prove that Benson's checking of stack-depth is outright inapposite or contradictory with the claimed limitation as to 'at the start of … determine if resources … needed by a block of code'.

Therefore, the rejections will stand as set forth in the Action.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please consult Examiner before using this number)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
July 27, 2004

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100